Прикладная математика (4 семестр)
Лабораторная работа 1
"Радиация"

Александров Даниил, Ершов Александр
Вариант №6

ITMO UNIVERSITY

1 Описание

Исследование методов одномерной минимизации

2 Цели

Реализовать методы одномерной минимизации и, в соответствии с вариантом, применить их для заданной функции на промежутке, где функция унимодальна

3 Цели

Реализовать методы одномерной минимизации, собрать данные о количестве итераций циклов (т.е. посмотреть на скорость сходимости), проанализировать количество вычислений функций, провести анализ ихменения интервала неопределенности, дать оценку эффективности методов

Часть I
Метод дихотомии

Метод относится с последовательным стратегиям. Задается начальный интервал неопределенности и требуемая точность. Алгоритм опирается на анализ функции в двух точках. Для этого от середины текущего интервала откладывается вправо и влево по $\frac{\delta}{2}$. Процесс поиска завершается, когда длина текущего интервала неопределенности меньше установленной точности.

1 Алгоритм

1. Зададим интервал неопределенности $L = [a_0, b_0]$, точность $\varepsilon > 0$
2. Пусть $k = 0$
3. Вычислим $y_k = \frac{a_k + b_k - \varepsilon}{2}$, $z_k = \frac{a_k + b_k + \varepsilon}{2}$, $f(y_k)$, $f(z_k)$
4. Сравним $f(y_k)$ и $f(z_k)$:
   Если $f(y_k) <= f(z_k)$, тогда $a_{k+1} = a_k$, $b_{k+1} = z_k$
   Иначе - $a_{k+1} = y_k$, $b_{k+1} = b_k$
5. Вычислим $|L_{2(k+1)}|$ и проверим условие окончания:
   Если $|L_{2(k+1)}| \leq \varepsilon$, тогда $x^* \approx \frac{a_k + b_k}{2}$, алгоритм завершаем.
   Иначе - $k = k + 1$ и переходим к шагу 2.

2 Сходимость

Характеристика уменьшения интервала неопределенности может быть описана как $R(N) = \frac{1}{2^{\frac{N}{2}}}$, где N - количество вычислений функции

Часть II

Метод золотого сечения

Метод относится с последовательным стратегиям. Задается начальный интервал неопределенности и требуемая точность. лгоритм опирается на анализ функции в двух точках. В качестве точек выбираются точки золотого сечения. Процесс поиска завершается, когда длина текущего интервала неопределенности меньше установленной точности.

1 Алгоритм

1. Зададим интервал неопределенности $L = [a_0, b_0]$, точность $\varepsilon > 0$
2. Пусть $k = 0$
3. Вычислим $y_0 = a_0 + \frac{3-\sqrt{5}}{2}(b_0 - a_0)$, $z_0 = a_0 + b_0 - y_0$
4. Вычислим $f(y_k)$, $f(z_k)$
5. Сравним $f(y_k)$ и $f(z_k)$:
   Если $f(y_k) <= f(z_k)$, тогда: $a_{k+1} = a_k$, $b_{k+1} = z_k$, $y_{k+1} = a_k + b_k - y_k$, $z_{k+1} = y_k$
   Иначе: $a_{k+1} = y_k$, $b_{k+1} = b_k$, $y_{k+1} = z_k$, $z_{k+1} = a_k + b_k - z_k$
6. Вычислим $\delta = a_{k+1} - b_{k+1}$ и проверим условие окончания:
   Если $|\delta| \leq \varepsilon$, тогда $x^* \approx \frac{a_{k+1}+b_{k+1}}{2}$, алгоритм завершаем.
   Иначе - $k = k + 1$ и переходим к шагу 4.

2 Сходимость

Характеристика уменьшения интервала неопределенности может быть описана как $R(N) = (0.618)^{N-1}$, где N - количество вычислений функции

Часть III

Метод Фибоначчи

Метод относится с последовательным стратегиям. Задается начальный интервал неопределенности и количество N вычислений функции. Алгоритм опирается на анализ функции в двух точках. Точки выбираются с использованием последовательности из $N+1$ чисел Фибоначчи. Процесс поиска завершается, когда длина текущего интервала неопределенности меньше установленной точности.

1 Алгоритм

1. Зададим интервал неопределенности $L = [a_0, b_0]$, точность $\varepsilon > 0$
2. Вычислим N так, чтобы $F_N \geq \frac{|L_0|}{\varepsilon}$, где $F_1..F_N$ - числа Фибоначчи
3. Пусть $k = 0$
4. Вычислим $y_0 = a_0 + \frac{F_{N-2}}{F_N}(b_0 - a_0)$, $z_0 = a_0 + \frac{F_{N-1}}{F_N}(b_0 - a_0)$
5. Вычислим $f(y_k)$, $f(z_k)$
6. Сравним $f(y_k)$ и $f(z_k)$:
   Если $f(y_k) <= f(z_k)$, тогда: $a_{k+1} = a_k$, $b_{k+1} = z_k$, $z_{k+1} = y_k$, $y_{k+1} = a_{k+1} + \frac{F_{N-k-3}}{F_{N-k-1}}(b_{k+1} - a_{k+1})$,
   Иначе: $a_{k+1} = y_k$, $b_{k+1} = b_k$, $y_{k+1} = z_k$, $z_{k+1} = a_{k+1} + \frac{F_{N-k-2}}{F_{N-k-1}}(b_{k+1} - a_{k+1})$
7. (a) если $k \neq N-3$, тогда $k = k+1$ и переходим к шагу 5
   (b) если $k = N-3$, тогда
      i. $y_{N-1} = y_{N-2} = z_{N-2}$
      ii. $z_{N-1} = y_{N-1} + \varepsilon$
      iii. Вычисляем $f(y_{N-1})$ и $f(z_{N-1})$:
      Если $f(y_{N-1}) <= f(z_{N-1})$: $a_{N-1} = a_{N-2}$, $b_{N-1} = z_{N-1}$,
      Иначе: $a_{N-1} = y_{N-1}$, $b_{N-1} = b_{N-2}$,
      iv. Остановка поиска, $x^* \approx \frac{a_{N-1}+b_{N-1}}{2}$

2 Сходимость

Характеристика уменьшения интервала неопределенности может быть описана как $R(N) = \frac{1}{F_N}$, где N - количество вычислений функции

Часть IV

Метод квадратичной интерполяции

Метод относится с последовательным стратегиям. Задается начальная точка и спомощью пробного шага находятся три точки так, чтобы они были как можно ближе к исходной точке минимума. В полученных точа вычисляются значения функции. Затем строится интерполяционный многочлен второго порядка (парабола), проходящий через найденные точки. В качестве приближения точки минимума берется абсцисса вершины данной параболы. Процесс поиска завершием, когда полученная точка минимума отличается от наилучшей из трех опорных точек не более чем на заданную величину.

1 Алгоритм

1. Зададим начальную точку x_1, величину шага $\Delta x > 0$, погрешности ε_1 и ε_2
2. Вычислим $x_2 = x_1 + \Delta x$
3. Вычислим $f_1 = f(x_1)$, $f_2 = f(x_2)$
4. Сравним f_1 и f_2:
   Если $f_1 <= f_2$, тогда: $x_3 = x_1 + 2\Delta x$
   Иначе: $x_3 = x_1 - \Delta x$
5. Вычислим $f_3 = f(x_3)$
6. Обозначим $f_{min} = min\{f_1, f_2, f_3\}$
   $x_{min} = x_i : f(x_i) = f_{min}$
7. Если $(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3 = 0$, тогда:
   результатом интерполяции является прямая, поэтому
   $x_1 = x_{min}$
   Переходим к шагу 2
8. Ищем вершину параболы
   $x_{mean} = \frac{1}{2}\frac{(x_2^2-x_3^2)f_1+(x_3^2-x_1^2)f_2+(x_1^2-x_2^2)f_3}{(x_2-x_3)f_1+(x_3-x_1)f_2+(x_1-x_2)f_3}$
   $f_{mean} = f(x_{mean})$
9. (a) если $|\frac{f_{min}-f_{mean}}{f_{mean}}| < \varepsilon_1$ и $|\frac{x_{min}-x_{mean}}{x_{mean}}| < \varepsilon_2$, тогда:
   Поиск завершен, $x^* \approx x_{mean}$
   (b) если $x_{mean} \in [x_1, x_3]$, тогда:
   Выбираем наилучшею точку (x_{min} или x_{mean}) и две точки по обе стороны от выбранной
   Упорядочиваем точки и значения функции в этих точках в естественном порядке
   Переход к шагу 6
   (c) если $x_{mean} \notin [x_1, x_3]$, тогда:
   Обозначаем $x_1 = x_{mean}$
   Переход к шагу 2

2 Сходимость

При неудачно выбранных параметрах шага и стартовой точки, поиск (особенно, если функция многомодальная), поиск может продолжаться бесконечно

Часть V

Комбинированный метод Брента

Метод относится с последовательным или иттеративным стратегиям. Выбираются три точки и в полученных точках вычисляются значения функции. Затем строится интерполяционный многочлен второго порядка (парабола), проходящий через найденные точки. В качестве приближения точки минимума берется абсцисса вершины данной параболы.На самом деле эту точку мы можем как принять, так и не принять, если она не попадает внутрь интервала [a, c] и отстоит от границ интервала не менее, чем на или если не отстоит от точки x не более, чем на половину от длины предпредыдущего шага. Процесс поиска завершён, когда полученная точка минимума отличается от наилучшей из трех опорных точек не более чем на заданную величину.

Если точка u отвергается, то следующая точка находится с помощью золотого сечения большего из интервалов [a, x] и [x, c]. Таким образом гарантируется, что комбинированный метод Брента будет работать не дольше, чем метод дихотомии.

Часть VI

Результаты

Ниже представлены сравнения методов по количеству операций, вычислений функции и итоговому значению на промежутке $[-5;0]$, где функция унимодальна.
Минимум на этом промежутке достигается в точке $x = -2.289$

1 Метод дихотомии

ε	Число итераций	Число вычислений функции	Результат
1	10	20	-2.287060
2	14	28	-2.288783
3	17	34	-2.288917
4	20	40	-2.288929
5	24	48	-2.288929
6	27	54	-2.288929

2 Золотое сечение

ε	Число итераций	Число вычислений функции	Результат
1	9	20	-2.287137
2	13	28	-2.290103
3	18	38	-2.288868
4	23	48	-2.288931
5	28	58	-2.288930
6	32	66	-2.288929

3 Метод Фибоначчи

ε	Число итераций	Число вычислений функции	Результат
1	6	14	-2.272727
2	11	24	-2.286885
3	16	34	-2.288856
4	21	44	-2.288887
5	25	52	-2.288931
6	30	62	-2.288930

4 Метод парабол

ε	Число итераций	Число вычислений функции	Результат
1	3	12	-2.28872
2	4	16	-2.288396
3	5	20	-2.288395
4	5	20	-2.288395
5	5	20	-2.288395
6	5	20	-2.288395

5 Метод Брента

ε	Число итераций	Число вычислений функции	Результат
1	5	6	-2.265248
2	7	8	-2.291896
3	9	10	-2.288983
4	12	13	-2.288960
5	15	16	-2.288930
6	19	20	-2.288928

Ниже представлены сравнения методов по количеству операций, вычислений функции и итоговому значению на промежутке $[-15; -5]$, где функция многомодальна.
Минимум на этом промежутке достигается в точке $x = -14.276$

1 Метод дихотомии

ε	Число итераций	Число вычислений функции	Результат
1	11	22	-8.097827
2	15	30	-8.096235
3	18	36	-8.096179
4	21	42	-8.096163
5	25	50	-8.096163
6	28	56	-8.096163

2 Золотое сечение

ε	Число итераций	Число вычислений функции	Результат
1	10	22	-8.090169
2	15	32	-8.096101
3	20	42	-8.096227
4	24	50	-8.096160
5	29	60	-8.096162
6	34	70	-8.096163

3 Метод Фибоначчи

ε	Число итераций	Число вычислений функции	Результат
1	8	18	-8.125
2	13	28	-8.091430
3	17	36	-8.096108
4	22	46	-8.096142
5	27	56	-8.096163
6	32	66	-8.096163

4 Метод парабол

ε	Число итераций	Число вычислений функции	Результат
1	2	8	-14.22621
2	2	8	-14.226212
3	3	12	-14.226703
4	4	16	-14.226744
5	5	20	-14.226748
6	6	24	-14.226748

5 Метод Брента

ε	Число итераций	Число вычислений функции	Результат
1	6	7	-8.090169
2	6	7	-8.100169
3	9	10	-8.096060
4	8	9	-8.096160
5	9	10	-8.096164
6	11	12	-8.096163

Часть VII

Вывод

В ходе исследования методов установлено, что наилучшей сходимостью обладаем метод парабол. Однако, при нейдачно выбранных начальных условий поиска, сходимость не гарантируется. Отметим, что лучшим из оставшихся является комбинированный метод Брента. Но в ходе уменьшения интервала неопределенности методом золотого сечения может произойти ситуация, что в оставшемся промежутке не окажется локального минимума.

С точки зрения модальности, наилучшим методом можно назвать снова метод парабол, но здесь стоит учесть, что начальная точка поиска была в окресности локального минимума.

Часть VIII

Приложение

Листинг 1: Используемые библиотеки

```
import numpy as np
import matplotlib.pyplot as plt
import math
```

Листинг 2: Метод дихотомии

```
def dichotomyMethod(f, a, b, p):
    e = 10 ** (-p)
    A = [a]
    B = [b]
    Y = []
    Z = []
    counter = 0
    f_counter = 0

    while True:
        Y.append((A[counter] + B[counter] - e) / 2)
        f_y_k = f(Y[counter])
        Z.append((A[counter] + B[counter] + e) / 2)
        f_z_k = f(Z[counter])
        if f_y_k <= f_z_k:
            A.append(A[counter])
            B.append(Z[counter])
        elif f_y_k > f_z_k:
            A.append(Y[counter])
            B.append(B[counter])
        if round(B[counter] - A[counter], p + 1) <= e:
            break
        counter += 1
        f_counter += 2

    x_min = (A[counter] + B[counter]) / 2

    print(f"Total operations performed: {counter}")
    print(f"The function was calculated: {f_counter} times")
    print(f"Minimun of f(x) on the interval [{a}, {b}] is {f(x_min)} at x = {x_min}")

    return x_min, np.array(A), np.array(B), counter, f_counter
```

Листинг 3: Метод дихотомии

```
def goldenRatioMethod(f, a, b, p):
    fi = (3 - math.sqrt(5)) / 2
    k = 0
    f_counter = 0
    e = 10 ** (-p)
    A = [a]
    B = [b]
    Y = [A[k] + fi * (B[k] - A[k])]
    Z = [A[k] + B[k] - Y[k]]

    while True:
        f_y_k = f(Y[k])
        f_z_k = f(Z[k])
        f_counter += 2
        if f_y_k <= f_z_k:
```

```
            A.append(A[k])
            B.append(Z[k])
            Y.append(A[k+1] + B[k+1] - Y[k])
            Z.append(Y[k])
        elif f_y_k > f_z_k:
            A.append(Y[k])
            B.append(B[k])
            Y.append(Z[k])
            Z.append(A[k+1] + B[k+1] - Z[k])
        if round(abs(A[k] - B[k]), p+1) <= e:
            break
        k += 1

    x_min = (A[k] + B[k]) / 2

    print(f"Total operations performed: {k}")
    print(f"The function was calculated: {f_counter} times")
    print(f"Minimun of f(x) on the interval [{a}, {b}] is {f(x_min)} at x = {x_min}")

    return x_min, np.array(A), np.array(B), k, f_counter
```

Листинг 4: Последовательность Фибоначчи

```
def createFib(a, b, e):
    k = int((b - a) / e)
    if k == 1:
        return [1]
    a = [1, 1]

    tmp = [1, 1]
    while k >= a[len(a)-1]:
        tmp[0], tmp[1] = tmp[1], tmp[0] + tmp[1]
        a.append(tmp[1])
    return a
```

Листинг 5: Метод Фибоначчи

```
def FibonacciMethod(f, a, b, p):
    e = 10 ** (-p)
    A = [a]
    B = [b]
    k = 0
    f_counter = 0
    F = createFib(a, b, e)
    n = len(F)-1
    Y = [A[k] + F[n - 2] / F[n] * (B[k] - A[k])]
    Z = [A[k] + F[n - 1] / F[n] * (B[k] - A[k])]

    while True:
        f_y_k = f(Y[k])
        f_z_k = f(Z[k])
        f_counter += 2
        if f_y_k <= f_z_k:
            A.append(A[k])
            B.append(Z[k])
            Z.append(Y[k])
            Y.append(A[k+1] + F[n-k-3]/F[n-k-1]*(B[k+1] - A[k+1]))
        elif f_y_k > f_z_k:
            A.append(Y[k])
            B.append(B[k])
            Y.append(Z[k])
            Z.append(A[k+1] + F[n-k-2]/F[n-k-1]*(B[k+1] - A[k+1]))
```

```
        if k != n - 3:
            k += 1
            continue
        elif k == n - 3:
            Y[n-2] = Y[n-3]
            Z[n-2] = Y[n-2] + e

            f_y = f(Y[n-2])
            f_z = f(Z[n-2])
            if f_y_k <= f_z_k:
                A[n-2] = A[n-3]
                B[n-2] = Z[n-2]
            elif f_y_k > f_z_k:
                A[n-2] = Y[n-2]
                B[n-2] = B[n-3]
            break


    x_min = (A[n-2] + B[n-2]) / 2

    print(f"Total operations performed: {k}")
    print(f"The function was calculated: {f_counter} times")
    print(f"Minimun of f(x) on the interval [{a}, {b}] is {f(x_min)} at x = {x_min}")

    return x_min, np.array(A), np.array(B), k, f_counter
```

Листинг 6: Метод парабол

```
def parabola_search(f, a, dx, p):
    X_mem = []
    X = [0 for i in range(3)]
    F = [0 for i in range(3)]

    X[0] = a
    e_1 = 10 ** (-p)
    e_2 = 10 ** (-p-1)
    x_res = a
    flag = False
    X_means = []
    counter = 0
    f_counter = 0

    while True:
        counter += 1
        if not flag:
            X[1]= X[0] + dx
            F[0] = f(X[0])
            F[1] = f(X[1])
            f_counter += 2
            if F[0] > F[1]:
                X[2] = X[0] + 2 * dx
            elif F[0] <= F[1]:
                X[2] = X[0] - dx
            F[2] = f(X[2])
            f_counter += 1
        X_mem.append(X)
        F_min = min(F)
        x_min = X[F.index(F_min)]

        numerator = F[0] * (X[1] ** 2 - X[2] ** 2) + F[1]
            * (X[2] ** 2 - X[0] ** 2) + F[2] * (X[0] ** 2 - X[1] ** 2)
        denominator = 2 * (F[0] * (X[1] - X[2]) + F[1]
            * (X[2] - X[0]) + F[2] * (X[0] - X[1]))
```

```
        if denominator == 0:
            X[0] = x_min
            continue

        x_mean = round(numerator / denominator, p+3)
        X_means.append(x_mean)
        F_mean = f(x_mean)
        f_counter += 1
        if abs((F_min - F_mean)/(F_mean + 1e-6)) <= e_1
            and abs((x_min - x_mean)/(x_mean + 1e-6)) <= e_2:
            x_res = x_mean
            break
        else:
            if X[0] <= x_mean <= X[2]:
                flag = True
                if X[0] < x_mean < X[1]:
                    if F_min < F_mean:
                        X[0], X[1], X[2] = x_min - dx, x_min, x_min + dx
                        F[0], F[1], F[2] = f(X[0]), f(X[1]), f(X[2])
                    else:
                        X[0], X[1], X[2] = x_mean - dx, x_mean, x_mean + dx
                        F[0], F[1], F[2] = f(X[0]), f(X[1]), f(X[2])
                elif X[1] < x_mean < X[2]:
                    X[0], X[1], X[2] = X[1], x_mean, X[2]
                    F[0], F[1], F[2] = f(X[0]), f(X[1]), f(X[2])
                f_counter += 3
                continue
            elif X[0] > x_mean or x_mean > X[2]:
                flag = False
                X[0] = x_mean
                continue

    print(f'Total operations performed: {counter}')
    print(f"The function was calculated: {f_counter} times")
    print(f'Minimum of f is at x = {x_res} and y = {f(x_res)}')

    return x_res, f(x_res), X_means, X_mem, counter, f_counter
```

Листинг 7: Метод Брента

```
def brents_method(f, a, c, p, eps):
    counter = 0
    f_counter = 0
    A, C = [a], [c]
    K = (3 - math.sqrt(5)) / 2
    x, w, v = (a + c) / 2, (a + c) / 2, (a + c) / 2
    tmp = f(x)
    f_counter += 1
    fx, fw, fv = tmp, tmp, tmp
    d, e = c - a, c - a

    while True:
        counter += 1
        g, e = e, d

        numerator = (w - x)**2 * (fw - fv) - (w - v)**2 * (fw - fx)
        denominator = 2 * ((w - x) * (fw - fv) - (w - v) * (fw - fx))
        u = w - numerator / (denominator + 1e-9)

        if a + eps <= u <= c - eps and abs(u - x) < g / 2:
            d = np.abs(u - x)
        else:
```

```
            if x < (c - a) / 2:
                u = x + K * (c - x)
                d = c - x
            else:
                u = x - K * (x - a)
                d = x - a

        if abs(u - x) < eps:
            u = x + np.sign(u - x) * eps

        fu = f(u)
        f_counter += 1
        if fu <= fx:
            if u >= x:
                a = x
            else:
                c = x
            A.append(a)
            C.append(c)
            v, w, x = w, x, u
            fv, fw, fx = fw, fx, fu
        else:
            if u >= x:
                c = u
            else:
                a = u
            A.append(a)
            C.append(c)
            if fu <= fw or w == x:
                v, w = w, u
                fv, fw = fw, fu
            elif fu <= fv or v == x or v == w:
                v = u
                fv = fu

        if round(abs(a - c), p + 1) <= 3*eps:
            break

    print(f'Total operations performed: {counter}')
    print(f"The function was calculated: {f_counter} times")
    print(f'Minimum of f is at y = {f(x)} and x = {x}')

    return x, np.array(A), np.array(C), counter, f_counter
```